(12) UK Patent Application (19) GB (11) 2 318 894 (13) A

(43) Date f A Publication 06.05.1998

(21) Application No 9716473.5

(22) Date of Filing 04.08.1997

(30) Priority Data
(31) 960753 (32) 29.10.1996 (33) IE

(71) Applicant(s)
Sportables Limited

(Incorporated in Ireland)

Howth House, Harbour Road, HOWTH, County Dublin, Ireland

(72) Inventor(s)
Neil Wilson
Patrick John Ratcliffe
Steven John Metzler

(74) Agent and/or Address for Service
Lloyd Wise, Tregear & Co
Commonwealth House, 1-19 New Oxford Street, LONDON, WC1A 1LW, United Kingdom

(51) INT CL[6]
G06F 13/16 13/368

(52) UK CL (Edition P)
G4A AFN AMB

(56) Documents Cited
EP 0130593 A2 EP 0032182 A1 WO 83/04117 A1

(58) Field of Search
UK CL (Edition O) G4A AFN AMB
INT CL[6] G06F 9/46 13/16 13/18 13/20 13/22 13/374 13/376
ONLINE: WPI, INSPEC, COMPUTER

(54) Controlling access by two computer processors to a shared resource

(57) Ensuring that only one of two processors has access to a shared resource, such as a memory, at any time, and resolving simultaneous access requests. A processor seeking access to the resource sets, 24, a corresponding lock flag at the resource, and then checks, 25, whether the other processor has, in the meantime, set its lock flag. If the second processor determines that the first processor's lock flag has not been set, it seizes control of the resource and sets a contention flag to a value indicating successful seizure; otherwise, the second processor abandons the attempt and sets the contention flag accordingly (fig. 2). Should the first processor determine, 30, that the second processor has set its lock flag while it set its own lock flag, the first processor seizes control of the resource, 35, or otherwise, 33, depending on the value of the contention flag set by the second processor. The resource is released on task completion (figs. 3 and 4).

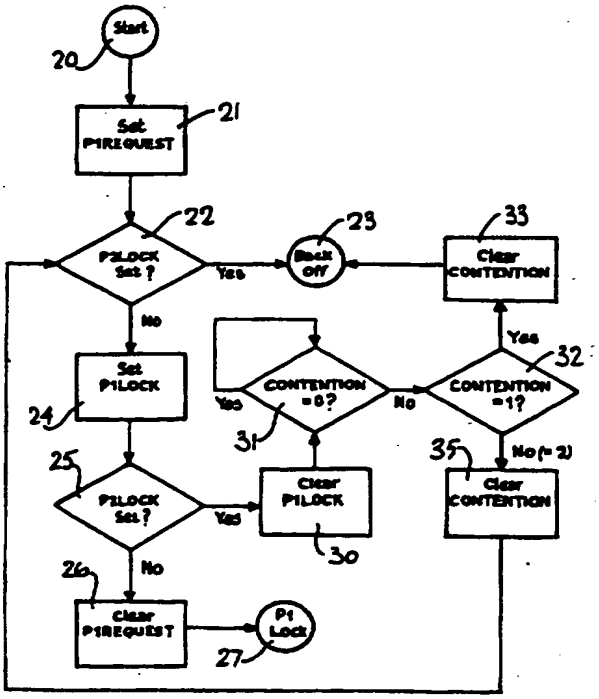

Fig.1

GB 2 318 894 A

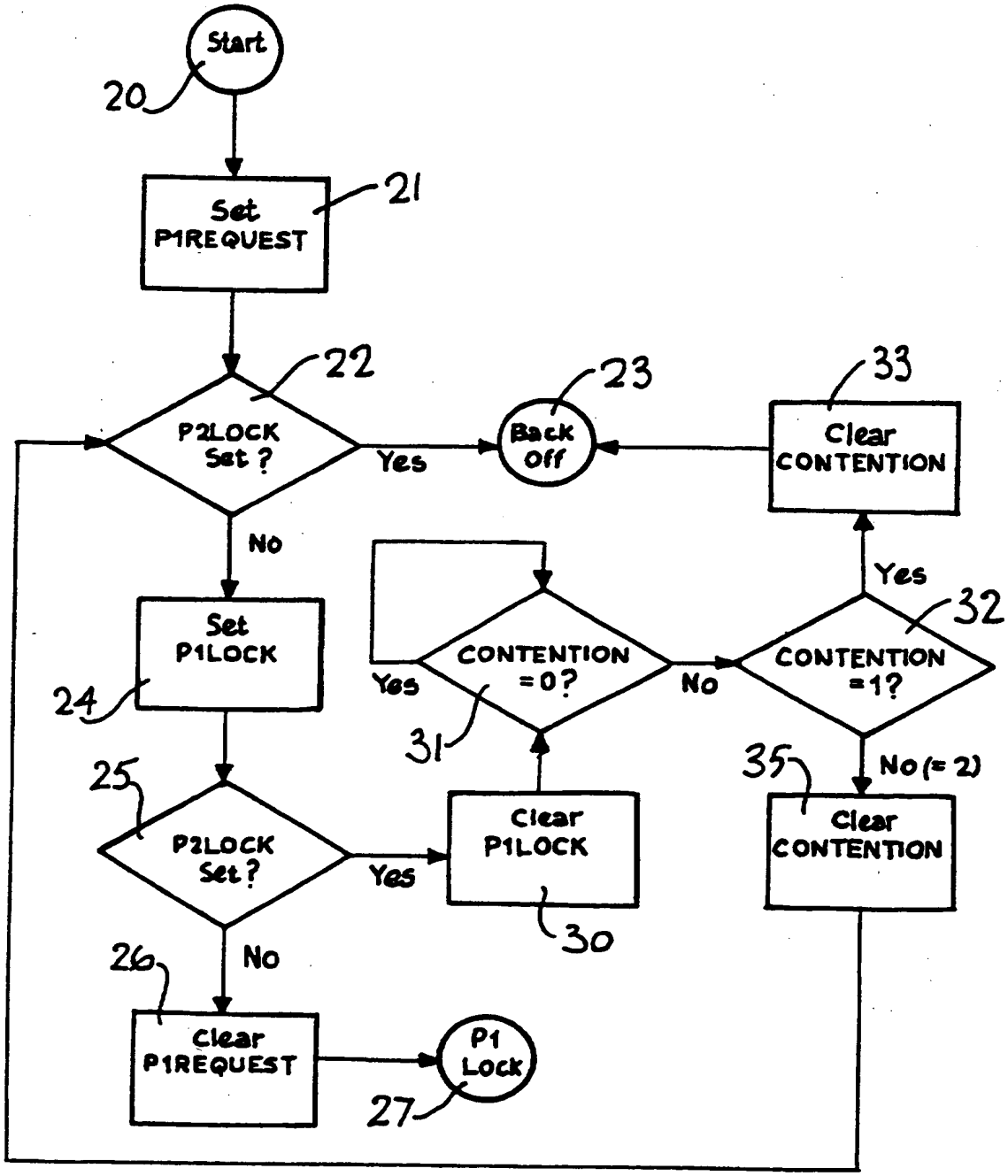
Start
20
Set P1REQUEST
21
P2LOCK Set ?
22
Yes
Back Off
23
Clear CONTENTION
33
No
Set P1LOCK
24
CONTENTION =0?
Yes
31
No
CONTENTION =1?
32
Yes
No (= 2)
35
P2LOCK Set?
25
Yes
Clear P1LOCK
30
Clear CONTENTION
No
26
Clear P1REQUEST
P1 Lock
27

Fig.1

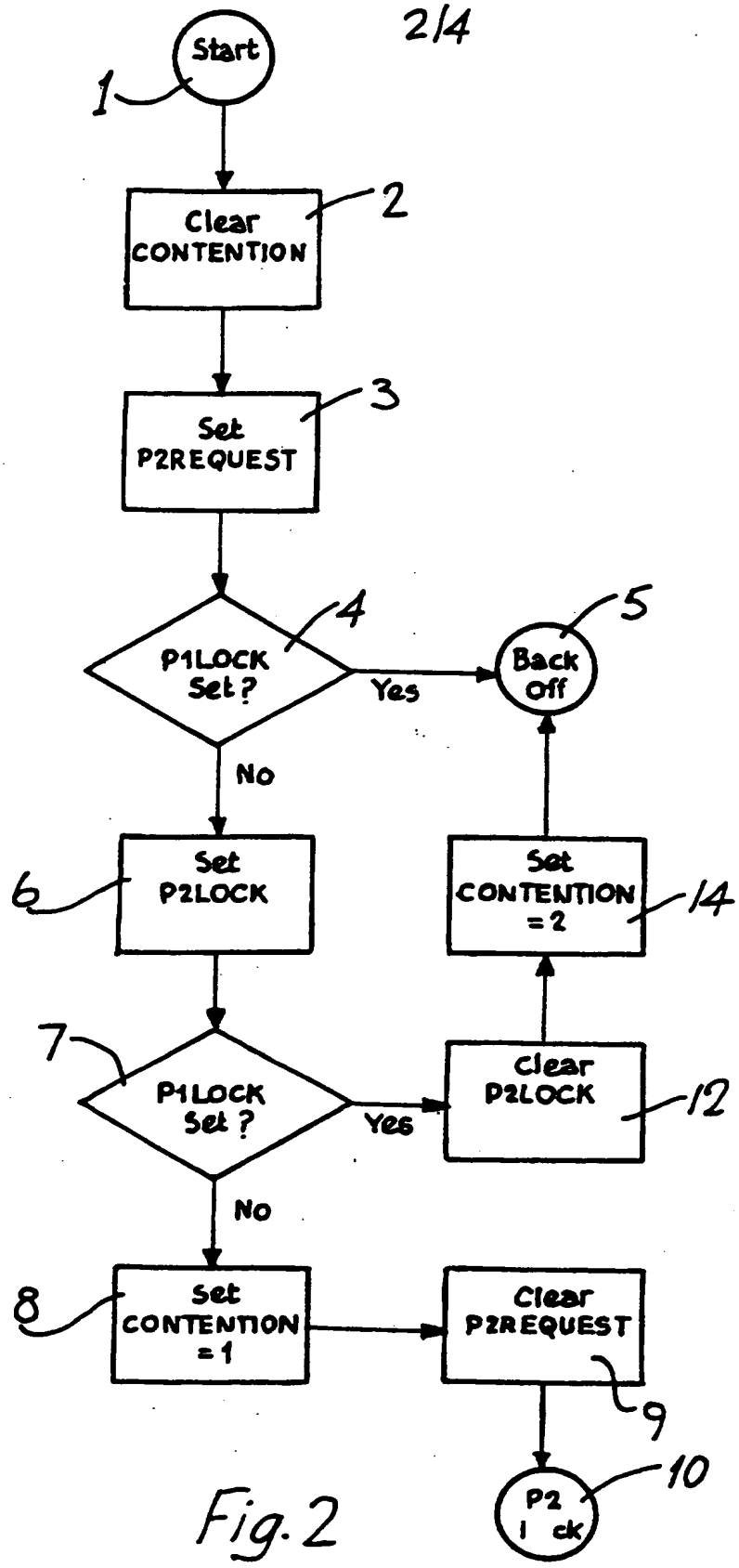
Start
1
Clear CONTENTION
2
Set P2REQUEST
3
P1LOCK Set?
4
Yes
Back off
5
No
Set P2LOCK
6
Set CONTENTION = 2
14
P1LOCK Set?
7
Yes
Clear P2LOCK
12
No
Set CONTENTION = 1
8
Clear P2REQUEST
9
P2 lock
10

Fig. 2

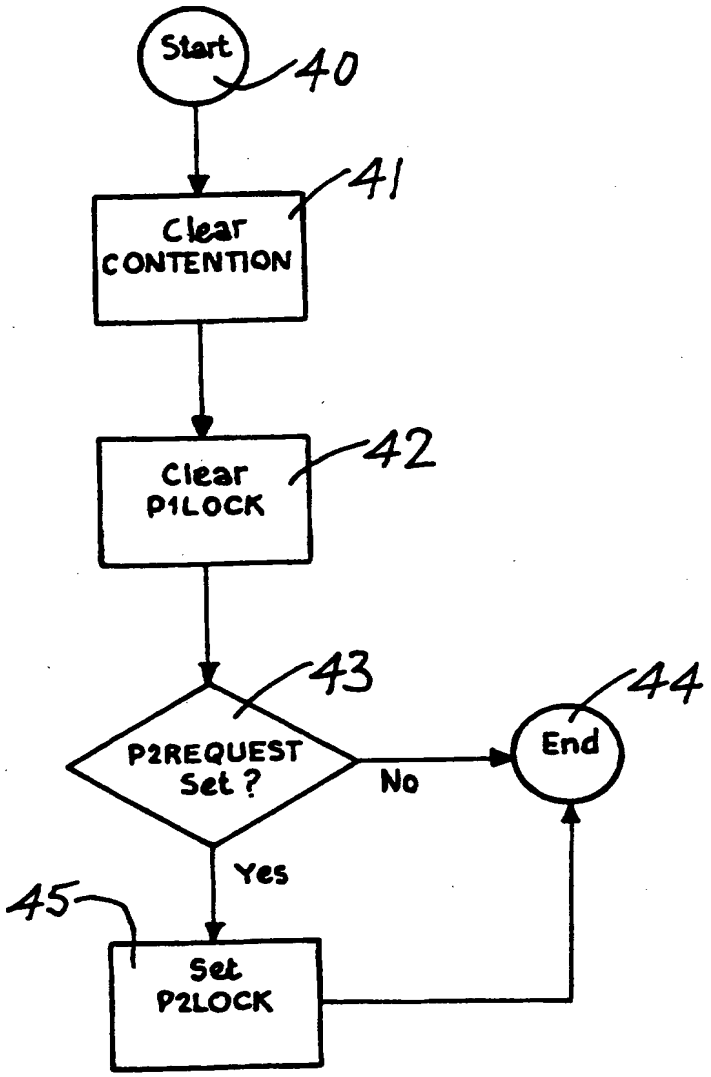
Start
40
41
Clear CONTENTION
42
Clear P1LOCK
43
P2REQUEST Set ?
No
44
End
Yes
45
Set P2LOCK

Fig.3

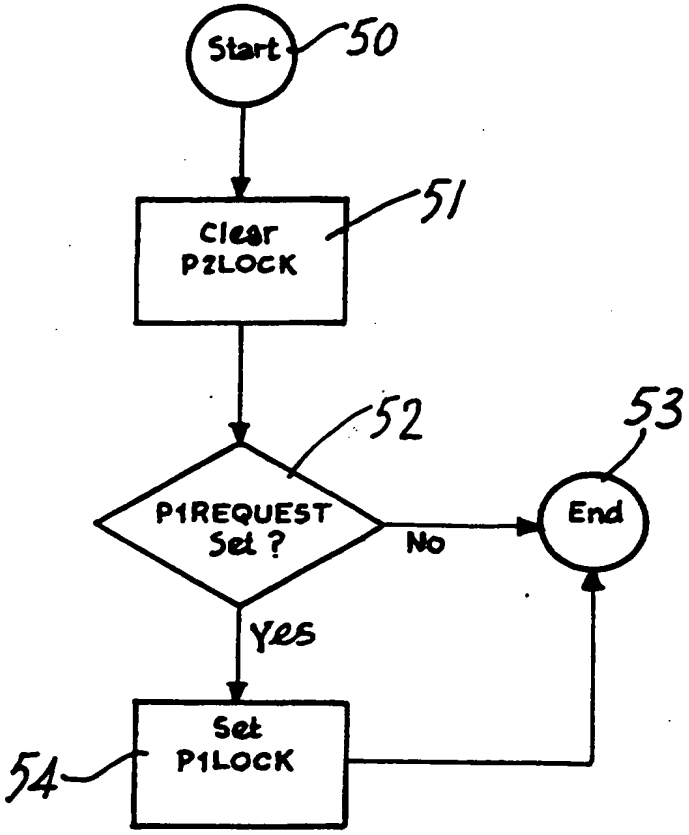
Start
50
Clear
P2LOCK
51
P1REQUEST
Set ?
52
No
End
53
Yes
Set
P1LOCK
54

Fig.4

"A method and apparatus for controlling access by two computer processors to a shared resource"

The present invention relates to a method and apparatus for controlling access by two computer processors, namely, a first processor and a second processor to a shared resource so that only one of the two processors has access to the resource at any one time. The invention also relates to a computer programme adapted for carrying out the method.

It is known to control access by two computer processors, for example, the central processor units of a computer, or the central processor units of respective computers to a shared resource, typically, a common memory resource using hardware, for example, bus locking mechanisms. However, where such hardware locking mechanisms are not provided, controlling access to a shared resource by two computer processors so that only one of the processors has access to the resource at any one time presents serious problems. Such problems arise in particular, when the processors execute instructions at disparate speeds, and/or where the processors are entirely different. Indeed, such problems arise even where the computer processors are similar, and operate at similar speeds.

There is therefore a need for a method for controlling access by two computer processors to a shared resource so that only one of the two processors has access to the resource at any one time.

The present invention is directed towards providing such a method, and also, to an apparatus for controlling access by the two computers to the shared resource. The invention is also directed to a computer programme adapted for carrying out the method.

According to the invention there is provided a method for controlling access by two computer processors, namely, a first processor and a second processor to a shared resource so that only one of the two processors has access to the resource at any one time, the method comprising the steps of:

causing each of the first and second processors to set a corresponding lock flag at the resource as the respective processors are seizing control of the resource, the first processor setting a first lock flag, and the second processor setting a second lock flag,

causing the second processor to set a contention flag at the resource to a first value as the second processor seizes control of the resource, and to a second value if the second processor fails to seize

control of the resourc ,

so that if the two processors are simultaneously attempting to seize control of the resource, and both processors have set their respective lock flags, the first processor can ascertain from the value of the contention flag if the second processor has successfully seized the resource.

Preferably, the first processor seizes control of the resource in response to the value of the contention flag. Advantageously, the first processor seizes control of the resource in response to the status of the second lock flag.

Preferably, the second processor seizes control of the resource in response to the status of the first lock flag.

In one embodiment of the invention the first processor prior to setting the first lock flag makes a first check to ascertain if the second lock flag has been set, and if the second flag has been set, the first processor abandons that attempt to seize control of the resource, if the second lock flag has not been set, the first processor sets the first lock flag.

Preferably, the first processor having set the first

lock flag makes a second check to ascertain if the second lock flag has be n s t, and if the second lock flag has not been set the first processor seizes control of the resource.

Advantageously, the first processor having detected as a result of the second check that the second lock flag has been set, clears the first lock flag and checks the contention flag, and continues to check the contention flag until it has been set to the first or second value.

In one embodiment of the invention the first processor on detecting that the contention flag has been set to the first value clears the contention flag and abandons that attempt to seize control of the resource.

In another embodiment of the invention the first processor on detecting the contention flag has been set to the second value clears the contention flag and proceeds to make a further check on the second lock flag, which further check corresponds to the first check on the second lock flag, and the first processor then continues to repeat the steps of the method after the first check on the second lock flag, depending on th results of the said further check.

Preferably, the s cond processor before commencing an attempt to seize control of the resource clears the contention flag.

In one embodiment of the invention the second processor prior to setting the second lock flag makes a first check to ascertain if the first lock flag has been set, and if so, the second processor abandons that attempt to seize control of the resource, if the first lock flag has not been set, the second processor sets the second lock flag.

In another embodiment of the invention the second processor having set the second lock flag makes a second check to ascertain if the first lock flag has been set, and if the first lock flag has not been set, the second processor sets the contention flag to the first value and seizes control of the resource.

In a further embodiment of the invention the second processor having determined as a result of the second check that the first lock flag has been set, the second processor clears the second lock flag and sets the contention flag to the second value and abandons that attempt to seize control of the resource.

In one embodiment of the invention th method furth r

compris s the step of causing ach of the first and second processors to set corr sponding request flag at the resource to indicate a request for access to the resource, the first processor setting a first request flag, and the second processor setting a second request flag.

In one embodiment of the invention the first processor on completing a task in the resource clears the contention flag and the first lock flag, for releasing control of the resource, and checks if the second request flag has been set. Preferably, the first processor on completing the task in the resource, and on determining that the second request flag has not been set terminates the method, and on determining that the second request flag has been set, the first processor sets the second lock flag for providing access to the resource by the second processor.

In another embodiment of the invention the second processor on completing a task in the resource clears the second lock flag for releasing control of the resource, and checks if the first request flag has been set.

Pr ferably, th s cond processor on completing the task in the resource, and on determining that th first

request flag has not been s t terminates the method, and on determining that the first request flag has been set sets the first lock flag for providing access to the resource by the first processor.

In one embodiment of the invention each processor clears the request flag as it seizes control of the resource.

Additionally, the invention provides apparatus for controlling access by two computer processors, namely, a first processor and a second processor to a shared resource so that only one of the two processors has access to the resource any one time, the apparatus comprising:

a means for permitting each of the first and second processors to set a corresponding lock flag at the resource as the respective processors are seizing control of the resource, the first processor setting a first lock flag, and the second processor setting a second lock flag,

a means for permitting the second processor to set a contention flag at the resource to a first value as the second processor seizes control of the resource, and to a second value as the second processor fails to seize control of the resource,

so that if the two processors are simultaneously

attempting to seize control of the resource, and both processors have set th ir resp ctive lock flags, th first processor can ascertain from the value of the contention flag if the second processor has successfully seized the resource.

In one embodiment of the invention the first processor comprises a first seizing means for seizing control of the resource, the first seizing means being responsive to the value of the contention flag. Preferably, the first seizing means is responsive to the status of the second lock flag.

In another embodiment of the invention the second processor comprises a second seizing means for seizing control of the resource, the second seizing means being responsive to the status of the first lock flag.

Additionally the invention comprises a computer programme for controlling access by two computer processors, namely, a first processor and a second processor to a shared resource so that only one of the two processors has access to the resource at any one time, the computer programme being adapted for carrying out the method according to the invention.

Furth r, the inv ntion provides a comput r programme

comprising a sub-routin for controlling access by two computer processors, namely, a first processor and a second processor to a shared resource so that only one of the two processors has access to the resource at any one time, the sub-routine being adapted for carrying out the method according to the invention.

The invention also provides a medium comprising a computer programme according to the invention.

The invention will be more clearly understood from the following description of a preferred embodiment thereof which is given by way of example only with reference to the accompanying drawings, in which:

Fig. 1 is a flow chart of a sub-routine of a computer programme which is loaded into one of a pair of computer processors for controlling access by the two computer processors to a shared resource, the sub-routine being part of a method according to the invention for controlling access to the shared resource,

Fig. 2 is a flow chart of a sub-routine of a computer programme forming another part of the m thod for controlling access to the shar d resource, the sub-routine being loaded into the

other of the pair of computer proc ssors,

Fig. 3 is a flow chart of a sub-routine of a computer programme which also forms part of the method according to the invention for controlling access by the two computers to the resource, the sub-routine being loaded into the computer processor into which the sub-routine of Fig. 1 is loaded, and

Fig. 4 is a flow chart of a sub-routine of a computer programme which also forms part of the method according to the invention for controlling access by the two computers to the resource, the sub-routine being loaded into the computer processor into which the sub-routine of Fig. 2 is loaded.

Referring to the drawings there is illustrated four sub-routines of a computer programme for carrying out a method according to the invention for controlling access by two computer processors to a shared resource, in this embodiment of the invention a shared memory resource. The sub-routines of Figs. 1 and 3 control one of the computer processors, which for convenience, will be referred to as th first processor, and the sub-routines of Figs. 2 and 4 control th other of the

computer processors, which for convenience will b referred to as the second processor. Before describing the sub-routines in detail with reference to Figs. 1 to 4, a broad outline of the method for controlling the two processors will be given.

Each processor on desiring access to the resource sets a corresponding request flag at the resource to indicate a request for access to the resource. The first processor sets a first request flag, and the second processor sets a second request flag. Each processor also sets a corresponding lock flag at the resource as the corresponding processor is seizing control of the resource. The first processor sets a first lock flag, while the second processor sets a second lock flag. The second processor sets a contention flag at the resource to a first value, in this case one, as the second processor seizes control of the resource for indicating to the first processor that the second processor has successfully seized the resource. The second processor sets the contention flag to a second value, in this case two, should the second processor fail to successfully seize control of the resource. By setting the contention flag to either one of the two values, the first processor if it is simultan ously attempting to seize control of the resource, and if both the first and second processors

have s t their respective first and second lock flags, the first processor can asc rtain from th value of the contention flag whether the second processor has successfully seized the resource or otherwise. Should the second processor not have successfully seized control of the resource, the first processor then proceeds to do so. Should the second processor have successfully seized control, the first processor abandons that attempt.

Turning now to Figs. 1 to 4, the four sub-routines of the computer programme will now be described. However, for convenience, the sub-routine of Fig. 2 will first be described. The sub-routine of Fig. 2 is carried out by the second processor in an attempt to seize control of the resource. Block 1 of the sub-routine commences the sub-routine which then moves to Block 2. Block 2 clears the contention flag to the value zero should it have been set. The sub-routine then moves to Block 3 which sets the second request flag to the value one, indicating to the first processor that the second processor is requesting access to the resource. The sub-routine having set the second request flag moves to Block 4 which makes a first check to ascertain if the first lock flag has been set by the first processor at the resource. If Block 4 asc rtains that the first lock flag has been set, the sub-routine moves to Block

5 which terminates the sub-routine, thereby abandoning that attempt by the second processor to gain access to the resource.

On the other hand, should Block 4 determine that the first lock flag has not been set, the sub-routine moves to Block 6 which sets the second lock flag of the second processor to the value one at the resource. This indicates to the first process that the second processor is attempting to seize control of the resource. The sub-routine then moves to Block 7 which makes a second check to ascertain if the first processor had set the first lock flag at the resource while the second processor was setting the second lock flag. Should Block 7 determine that the first lock flag has not been set, the sub-routine moves to Block 8 which seizes control of the resource and locks the resource for the second processor, and also sets the contention flag to the first value, namely one, thereby indicating to the first processor that the second processor has successfully seized control of the resource. The sub-routine then moves to Block 9 which clears the second request flag to the value zero, and in turn moves to Block 10 which initiates the relevant programme to carry out the desired task at the resource.

Should Block 7 determine that while the second lock flag of the second processor was being set th first lock flag of the first processor had also been set, then the sub-routine moves to Block 12. Block 12 clears the second lock flag to the value zero, and then moves the sub-routine to Block 14, which sets the contention flag at the resource to the second value, namely two. This, thus indicates to the first processor that the second processor was unsuccessful in seizing the resource. The sub-routine then moves to Block 5 which terminates the sub-routine and abandons that attempt to seize control of the resource.

Referring now to Fig. 1 the sub-routine for controlling the first processor during an attempt to seize control of the resource will now be described. Block 20 starts the sub-routine which immediately moves to Block 21 which sets the first request flag of the first processor at the resource to the value one to indicate to the second processor that the first processor requires access to the resource. The sub-routine then moves to Block 22 which makes a first check to ascertain if the second lock flag has been set by the second processor. If so, the sub-routine moves to Block 23 which terminates the sub-routine, thereby abandoning that attempt to gain access to the resourc .

Should Block 22 determine that the second lock flag has not been set, the sub-routine moves to Block 24 which sets the first lock flag of the first processor at the resource to the value one. The sub-routine then moves to Block 25 which makes a second check to ascertain if the second lock flag had been set while the first lock flag was being set. Should Block 25 determine that the second lock flag had not been set, the sub-routine moves to Block 26 which seizes control of the resource and locks the resource for the first processor, and also clears the first request flag to zero. The sub-routine then moves to Block 27 which calls up the appropriate computer programme to carry out the desired task at the resource.

Should Block 25 determine that the second lock flag had been set, the sub-routine moves to Block 30 which clears the first lock flag. The sub-routine then moves to Block 31 which checks the value of the contention flag, which is set by the second processor. Should Block 31 determine that the contention flag is set to the value zero, in other words, is still cleared, the sub-routine remains in a loop until the value of the contention flag is not zero. At which stage, the sub-routine moves to Block 32. Block 32 checks if the value of th cont ntion flag is one, indicating that the second processor successfully seized control of th

resourc , and if Block 32 d termin s that the value of the contention flag is one the sub-routine moves to Block 33 which clears the value of the contention flag to zero and returns the sub-routine to Block 23 which terminates the sub-routine, thereby abandoning that attempt to seize control of the resource.

Should Block 32 determine that the value of the contention flag is not one, in other words, it must be two, the sub-routine moves to Block 35 which clears the value of the contention flag to zero, and returns the sub-routine to Block 22 to make a further check on the second lock flag, which is the equivalent to the first check on the second lock flag, and the sub-routine then continues through the remainder of the steps of the sub-routine depending on the state of the second lock flag.

Referring now to Fig. 3 the sub-routine of the computer programme by which the first processor releases the resource on the first processor having completed the task will now be described. Block 40 starts the sub-routine which then moves to Block 41. Block 41 clears the contention flag at the resource to zero, and the sub-routine moves to Block 42 which clears the first lock flag at th resource to zero, thereby releasing control of th resource. Th sub-routine th n moves to

Block 43 which checks if the second request flag had been set, indicating a request by the second processor to gain access to the resource. Should Block 43 determine that the second flag had not been set, the sub-routine moves to Block 44 which terminates the sub-routine. On the other hand, should Block 43 determine that the second request flag had been set, the sub-routine moves to Block 45 which sets the second lock flag of the second processor at the resource to one, and sends a message to the second processor that its second lock flag has been set. The sub-routine then moves to Block 44 which ends the routine.

Turning now to Fig. 4 the sub-routine of the computer programme by which the second processor releases the resource on completing the task at the resource will now be described. Block 50 starts the sub-routine which immediately moves to Block 51. Block 51 clears the second lock flag to zero, thereby releasing control of the resource. The sub-routine then moves to Block 52, which checks if the first request flag of the first processor has been set indicating a request to gain access to the resource. On Block 52 determining that the first request flag has not been set, the sub-routine moves to Block 53 which terminates the sub-routine. On the other hand, should Block 52 d termine that the first request flag had been s t, the sub-

routine moves to Block 54 which sets the first lock flag of the first processor at the resource to one, and sends a message to the first processor indicating that its first lock flag has been set. The sub-routine then moves to Block 53 which terminates the sub-routine.

It is desirable that both the first and second processors should be permitted to execute the sub-routines of Figs. 1 and 2 with their respective interrupts disabled so that arbitration carried out by the sub-routines is not preempted. This, in general, is important since the first processor which sits in a loop at Block 31 awaiting the contention flag to be set to either the first or second values to ascertain the course of action taken by the second processor.

CLAIMS

1. A method for controlling access by two computer processors, namely, a first processor and a second processor to a shared resource so that only one of the two processors has access to the resource at any one time, the method comprising the steps of:

causing each of the first and second processors to set a corresponding lock flag at the resource as the respective processors are seizing control of the resource, the first processor setting a first lock flag, and the second processor setting a second lock flag,

causing the second processor to set a contention flag at the resource to a first value as the second processor seizes control of the resource, and to a second value if the second processor fails to seize control of the resource,

so that if the two processors are simultaneously attempting to seize control of the resource, and both processors have set their respective lock flags, the first processor can ascertain from the value of the contention flag if the second processor has successfully seized the resource.

2. A method as claimed in Claim 1 in which the first processor seizes control of the resource in response to th value of th cont ntion flag.

3. A method as claimed in Claim 1 or 2 in which the first processor seizes control of the r source in response to the status of the second lock flag.

4. A method as claimed in any preceding claim in which the second processor seizes control of the resource in response to the status of the first lock flag.

5. A method as claimed in any preceding claim in which the first processor prior to setting the first lock flag makes a first check to ascertain if the second lock flag has been set, and if the second flag has been set, the first processor abandons that attempt to seize control of the resource, if the second lock flag has not been set, the first processor sets the first lock flag.

6. A method as claimed in Claim 5 in which the first processor having set the first lock flag makes a second check to ascertain if the second lock flag has been set, and if the second lock flag has not been set the first processor seizes control of the resource.

7. A method as claimed in Claim 6 in which the first proc ssor having d tected as a r sult of th s cond check that the second lock flag has b en s t, cl ars

the first lock flag and checks the contention flag, and continues to check the contention flag until it has been set to the first or second value.

8. A method as claimed in any preceding claim in which the first processor on detecting that the contention flag has been set to the first value clears the contention flag and abandons that attempt to seize control of the resource.

9. A method as claimed in any preceding claim in which the first processor on detecting the contention flag has been set to the second value clears the contention flag and proceeds to make a further check on the second lock flag, which further check corresponds to the first check on the second lock flag, and the first processor then continues to repeat the steps of the method after the first check on the second lock flag, depending on the results of the said further check.

10. A method as claimed in any preceding claim in which the second processor before commencing an attempt to seize control of the resource clears the contention flag.

11. A method as claimed in any preceding claim in

which th s cond processor prior to setting th second lock flag makes a first ch ck to ascertain if the first lock flag has been set, and if so, the second processor abandons that attempt to seize control of the resource, if the first lock flag has not been set, the second processor sets the second lock flag.

12. A method as claimed in Claim 11 in which the second processor having set the second lock flag makes a second check to ascertain if the first lock flag has been set, and if the first lock flag has not been set, the second processor sets the contention flag to the first value and seizes control of the resource.

13. A method as claimed in Claim 12 in which the second processor having determined as a result of the second check that the first lock flag has been set, the second processor clears the second lock flag and sets the contention flag to the second value and abandons that attempt to seize control of the resource.

14. A method as claimed in any preceding claim in which the method further comprises the step of causing each of the first and second processors to set corresponding request flag at the resource to indicate a request for acc ss to the resource, the first proc ssor setting a first request flag, and the second

processor setting a second request flag.

15. A method as claimed in Claim 14 in which the first processor on completing a task in the resource clears the contention flag and the first lock flag, for releasing control of the resource, and checks if the second request flag has been set.

16. A method as claimed in Claim 15 in which the first processor on completing the task in the resource, and on determining that the second request flag has not been set terminates the method, and on determining that the second request flag has been set, the first processor sets the second lock flag for providing access to the resource by the second processor.

17. A method as claimed in any of Claims 14 to 16 in which the second processor on completing a task in the resource clears the second lock flag for releasing control of the resource, and checks if the first request flag has been set.

18. A method as claimed in Claim 17 in which the second processor on completing the task in the resource, and on determining that the first request flag has not been set terminates the method, and on determining that the first r qu st flag has been set

sets th first lock flag for providing access to the resourc by the first proc ssor.

19. A method as claimed in any of Claims 14 to 18 in which each processor clears the request flag as it seizes control of the resource.

20. A method for controlling access by two computer processors, namely, a first processor and a second processor to a shared resource so that only one of the two processors has access to the resource at any one time, the method being substantially as described herein with reference to and as illustrated in the accompanying drawings.

21. Apparatus for controlling access by two computer processors, namely, a first processor and a second processor to a shared resource so that only one of the two processors has access to the resource any one time, the apparatus comprising:

a means for permitting each of the first and second processors to set a corresponding lock flag at the resource as the respective processors are seizing control of the resource, the first processor setting a first lock flag, and the second processor setting a second lock flag,

a means for permitting the second processor to set

a contention flag at the resource to a first valu as the second processor seizes control of the resource, and to a second value as the second processor fails to seize control of the resource,

so that if the two processors are simultaneously attempting to seize control of the resource, and both processors have set their respective lock flags, the first processor can ascertain from the value of the contention flag if the second processor has successfully seized the resource.

22. Apparatus as claimed in Claim 21 in which the first processor comprises a first seizing means for seizing control of the resource, the first seizing means being responsive to the value of the contention flag.

23. Apparatus as claimed in Claim 21 or 22 in which the first seizing means is responsive to the status of the second lock flag.

24. Apparatus as claimed in any of Claims 21 to 23 in which the second processor comprises a second seizing means for seizing control of the resource, the second seizing means being responsive to the status of the first lock flag.

25. Apparatus for controlling access by two comput r processors, namely, a first processor and a second processor to a shared resource so that only one of the two processors has access to the resource at any one time, the apparatus being substantially as described herein with reference to and as illustrated in the accompanying drawings.

26. A computer programme for controlling access by two computer processors, namely, a first processor and a second processor to a shared resource so that only one of the two processors has access to the resource at any one time, the computer programme being adapted for carrying out the method of any of Claims 1 to 20.

27. A computer programme comprising a sub-routine for controlling access by two computer processors, namely, a first processor and a second processor to a shared resource so that only one of the two processors has access to the resource at any one time, the sub-routine being adapted for carrying out the method of any of Claims 1 to 20.

28. A computer programme for controlling access by two computer processors, namely, a first processor and a second processor to a shared resource so that only one of the two proc ssors has access to th r sourc at any

one time, th computer programm being substantially as described herein with reference to and as illustrated in the accompanying drawings.

29. A medium comprising a computer programme as claimed in any of Claims 26 to 28.

The Patent Office

28

Application No: GB 9716473.5
Claims searched: 1-25

Examiner: Melanie Jennings
Date f search: 20 October 1997

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.O): G4A (AFN, AMB)

Int Cl (Ed.6): G06F 9/46, 13/16, 13/18, 13/20, 13/22, 13/374, 13/376

Other: Online: WPI, INSPEC, COMPUTER

Documents considered to be relevant:

Category	Identity of document and relevant passage		Relevant to claims
A	EP 0130593 A2	(HONEYWELL INFORMATION SYSTEMS), see pages 78-80 and fig. 10	
A	EP 0032182 A1	(HONEYWELL INFORMATION SYSTEMS), see pages 1-4	
A	WO 83/04117 A1	(WESTERN ELECTRIC), see pages 1-6	

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date f this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.